Burn

Generated by Doxygen 1.8.11

Contents

1	burn	1												1
2	Nam	espace I	ndex											3
	2.1	Package	es					 	 	 	 	 	 	 3
3	Hiera	archical I	ndex											5
	3.1	Class Hi	erarchy					 	 	 	 	 	 	 5
4	Clas	s Index												7
	4.1	Class Li	st					 	 	 	 	 	 	 7
5	File	Index												9
	5.1	File List						 	 	 	 	 	 	 9
6	Nam	espace [Oocumer	ntation										11
	6.1	burn Na	mespace	Referer	nce .			 	 	 	 	 	 	 11
	6.2	burn.bur	n Names	pace Re	eferenc	е		 	 	 	 	 	 	 11
		6.2.1	Variable I	Docume	ntation			 	 	 	 	 	 	 11
			6.2.1.1	filenam	ne			 	 	 	 	 	 	 11
			6.2.1.2	level				 	 	 	 	 	 	 11
	6.3	burn.hel	pers Nan	nespace	Refere	ence		 	 	 	 	 	 	 12
		6.3.1	Function	Docume	entation	١		 	 	 	 	 	 	 12
			6.3.1.1	setbloo	king(fd	, state)	 	 	 	 	 	 	 12
	6.4	burn.net	_proc Na	ımespac	e Refe	rence		 	 	 	 	 	 	 12
		6.4.1	Variable I	Docume	ntation			 	 	 	 	 	 	 12
			6.4.1.1	HOST				 	 	 	 	 	 	 12
			6.4.1.2	PORT				 	 	 	 	 	 	 12
	6.5	burn.pro	to Name	space R	eferenc	ce .		 	 	 	 	 	 	 13
	6.6	hurn sne	ec proc N	Jamesna	ace Ref	erenci	<u></u>							13

iv CONTENTS

7	Clas	s Docu	mentation	15
	7.1	burn.b	urn.Burn Class Reference	15
		7.1.1	Detailed Description	15
		7.1.2	Constructor & Destructor Documentation	15
			7.1.2.1init(self)	15
		7.1.3	Member Function Documentation	16
			7.1.3.1enter(self)	16
			7.1.3.2exit(self, exc_type, exc_value, traceback)	16
			7.1.3.3 dispatch_net_msg(self, msg)	16
			7.1.3.4 dispatch_spec_msg(self, msg)	17
			7.1.3.5 run(self)	17
		7.1.4	Member Data Documentation	17
			7.1.4.1 fdn	17
			7.1.4.2 fds	18
			7.1.4.3 n	18
			7.1.4.4 running	18
			7.1.4.5 s	18
	7.2	burn.s	pec_proc.GpsThread Class Reference	18
		7.2.1	Detailed Description	19
		7.2.2	Constructor & Destructor Documentation	19
			7.2.2.1init(self, event)	19
		7.2.3	Member Function Documentation	19
			7.2.3.1 altitude(self)	19
			7.2.3.2 eps(self)	20
			7.2.3.3 epv(self)	20
			7.2.3.4 epx(self)	20
			7.2.3.5 epy(self)	20
			7.2.3.6 latitude(self)	20
			7.2.3.7 longitude(self)	20
			7.2.3.8 run(self)	21

CONTENTS

		7.2.3.9 sp	peed(self)	2	21
		7.2.3.10 tin	me(self)	2	21
	7.2.4	Member Da	ta Documentation	2	21
		7.2.4.1 g _l	psd	2	21
		7.2.4.2 la	st_alt	2	22
		7.2.4.3 la	ast_eps	2	22
		7.2.4.4 la	st_epv	2	22
		7.2.4.5 la	st_epx	2	22
		7.2.4.6 la	st_epy	2	22
		7.2.4.7 la	ast_lat	2	22
		7.2.4.8 la	st_lon	2	22
		7.2.4.9 la	st_speed	2	22
		7.2.4.10 la	ast_time	2	22
7.3	burn.pı	roto.Message	Class Reference	2	23
	7.3.1	Detailed De	scription	2	23
	7.3.2	Constructor	& Destructor Documentation	2	23
		7.3.2.1	_init(self, command=", arguments={})	2	23
	7.3.3	Member Da	ta Documentation	2	23
		7.3.3.1 a	rguments	2	23
		7.3.3.2 co	ommand	2	24
7.4	burn.ne	et_proc.NetP	roc Class Reference	2	24
	7.4.1	Detailed De	scription	2	24
	7.4.2	Constructor	& Destructor Documentation	2	25
		7.4.2.1	_init(self, fd)	2	25
	7.4.3	Member Fu	nction Documentation	2	25
		7.4.3.1 di	ispatch_ctrl_msg(self, msg)	2	25
		7.4.3.2 di	ispatch_net_msg(self)	2	26
		7.4.3.3 is	s_running(self)	2	26
		7.4.3.4 ru	un(self)	2	27
	7.4.4	Member Da	ta Documentation	2	27

vi

		7.4.4.1	addr	27
		7.4.4.2	buffer	27
		7.4.4.3	$fd \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	28
		7.4.4.4	sock	28
7.5	object	Class Refe	erence	28
7.6	burn.s	pec_proc.S	SessionThread Class Reference	28
	7.6.1	Detailed	Description	28
	7.6.2	Construc	tor & Destructor Documentation	29
		7.6.2.1	init(self, event, target, msg)	29
	7.6.3	Member	Function Documentation	29
		7.6.3.1	run(self)	29
7.7	burn.s	pec_proc.S	SpecProc Class Reference	30
	7.7.1	Detailed	Description	30
	7.7.2	Construc	tor & Destructor Documentation	31
		7.7.2.1	init(self, fd)	31
	7.7.3	Member	Function Documentation	31
		7.7.3.1	dispatch(self, msg)	31
		7.7.3.2	reset_acquisition(self)	32
		7.7.3.3	run(self)	32
		7.7.3.4	run_acquisition(self, msg, session_index)	33
		7.7.3.5	run_acquisition_once(self, req_msg, session_index)	33
		7.7.3.6	save_acquisition(self, msg, session_index)	34
		7.7.3.7	save_acquisition_as_chn(self, sd, msg, session_index)	35
		7.7.3.8	send_msg(self, msg)	35
		7.7.3.9	stabilize_probe(self, voltage, coarse_gain, fine_gain)	36
	7.7.4	Member	Data Documentation	36
		7.7.4.1	dtb	36
		7.7.4.2	$fd \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	36
		7.7.4.3	gps_client	36
		7.7.4.4	gps_stop	36
		7.7.4.5	group	37
		7.7.4.6	input	37
		7.7.4.7	running	37
		7.7.4.8	send_lock	37
		7.7.4.9	session	37
		7.7.4.10	session_stop	37

Vi
VI

8	File	Documentation	39
	8.1	/home/drb/dev/py/burn/initpy File Reference	39
	8.2	/home/drb/dev/py/burn/burn.py File Reference	39
	8.3	/home/drb/dev/py/burn/helpers.py File Reference	39
	8.4	/home/drb/dev/py/burn/net_proc.py File Reference	40
	8.5	/home/drb/dev/py/burn/proto.py File Reference	40
	8.6	/home/drb/dev/py/burn/README.md File Reference	40
	8.7	/home/drb/dev/py/burn/spec_proc.py File Reference	40
Inc	dex		41

burn

Daemon running on Raspberry Pi (archlinuxarm), controlling a Canberra Osprey gamma detector and a G-Star IV GPS device.

Acquisitions and measurements are merged, saved to disk and optionally transferred over a TCP connection to the controlling application, crash.

Dependencies:

• Canberra Osprey SDK V1.0.1

This software is part of a drone project at Norwegian Radiation Protection Authority (NRPA)

2 burn

Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

burn	
burn.burn	
burn.helpers	
burn.net_proc	
burn.proto	
burn.spec proc	

4 Namespace Index

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

n.burn.Burn	15
ect	28
burn.proto.Message	23
ead	
burn.spec_proc.GpsThread	
burn.spec_proc.SessionThread	. 28
cess	
burn.net_proc.NetProc	. 24
burn.spec_proc.SpecProc	. 30

6 Hierarchical Index

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ırn.burn.Burn	. 15
ırn.spec_proc.GpsThread	. 18
ırn.proto.Message	. 23
ırn.net_proc.NetProc	. 24
o <mark>ject</mark>	. 28
ırn.spec_proc.SessionThread	. 28
ırn.spec proc.SpecProc	. 30

8 Class Index

File Index

5.1 File List

Here is a list of all files with brief descriptions:

/home/drb/dev/py/burn/initpy	39
/home/drb/dev/py/burn/burn.py	39
/home/drb/dev/py/burn/helpers.py	39
/home/drb/dev/py/burn/net_proc.py	40
/home/drb/dev/py/burn/proto.py	40
/home/drb/dev/py/burn/spec_proc.py	40

10 File Index

Namespace Documentation

6.1 burn Namespace Reference

Namespaces

- burn
- helpers
- net_proc
- proto
- spec_proc

6.2 burn.burn Namespace Reference

Classes

• class Burn

Variables

- filename
- level

6.2.1 Variable Documentation

6.2.1.1 burn.burn.filename

Definition at line 125 of file burn.py.

6.2.1.2 burn.burn.level

Definition at line 125 of file burn.py.

6.3 burn.helpers Namespace Reference

Functions

• def setblocking (fd, state)

6.3.1 Function Documentation

6.3.1.1 def burn.helpers.setblocking (fd, state)

```
Set the blocking state of a file descriptor
```

Definition at line 22 of file helpers.py.

```
22 def setblocking(fd, state):
23     """
24     Set the blocking state of a file descriptor
25     """
26     flags = fcntl.fcntl(fd, fcntl.F_GETFL)
27     if state:
28         fcntl.fcntl(fd, fcntl.F_SETFL, flags & ~os.O_NONBLOCK)
29     else:
30         fcntl.fcntl(fd, fcntl.F_SETFL, flags | os.O_NONBLOCK)
```

6.4 burn.net_proc Namespace Reference

Classes

• class NetProc

Variables

```
• string HOST = "
```

• int PORT = 7000

6.4.1 Variable Documentation

6.4.1.1 string burn.net_proc.HOST = "

Definition at line 25 of file net_proc.py.

6.4.1.2 int burn.net_proc.PORT = 7000

Definition at line 26 of file net_proc.py.

6.5 burn.proto Namespace Reference

Classes

• class Message

6.6 burn.spec_proc Namespace Reference

Classes

- class GpsThread
- class SessionThread
- class SpecProc

Class Documentation

7.1 burn.burn.Burn Class Reference

Public Member Functions

- def __init__ (self)
- def run (self)
- def dispatch_net_msg (self, msg)
- def dispatch_spec_msg (self, msg)
- def __enter__ (self)
- def __exit__ (self, exc_type, exc_value, traceback)

Public Attributes

- running
- fds
- fdn
- s
- n

7.1.1 Detailed Description

Definition at line 30 of file burn.py.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 def burn.burn.Burn.__init__ (self)

initialize main controller

Definition at line 32 of file burn.py.

```
def __init__(self):
32
            initialize main controller
34
35
36
            self.running = False
            # Create pipes for message passing between gps_proc and spec_proc
38
39
            fds_pass, self.fds = Pipe()
40
            fdn_pass, self.fdn = Pipe()
41
            # Make file descriptors non-blocking
42
           setblocking(self.fds, 0)
43
           setblocking(self.fdn, 0)
44
45
46
           # Create and start child processes
           self.s = SpecProc(fds_pass)
self.n = NetProc(fdn_pass)
47
48
            self.s.start()
49
50
           self.n.start()
           # Close unused file descriptors
# The child processes inherits *all* parent descriptors (FIXME)
53
54
            fds_pass.close()
55
            fdn_pass.close()
56
```

7.1.3 Member Function Documentation

```
7.1.3.1 def burn.burn.Burn._enter__ ( self )
```

Definition at line 106 of file burn.py.

```
106 def __enter__(self):
107 return self
108
```

7.1.3.2 def burn.burn.Burn._exit_ (self, exc_type, exc_value, traceback)

Definition at line 109 of file burn.py.

7.1.3.3 def burn.burn.Burn.dispatch_net_msg (self, msg)

Function to handle messages from the network

Definition at line 81 of file burn.py.

```
def dispatch_net_msg(self, msg):
83
            Function to handle messages from the network
84
            if not msg:
8.5
86
            if msg.command == 'ping':
88
                msg.command = 'ping_ok'
            self.fdn.send(msg)
elif msg.command == 'close':
89
90
                 # Ground control has requested a close down
91
                self.fds.send(msg) # Notify spectrometer
msg.command = 'close_ok' # Convert the message to a response
92
93
                self.fdn.send(msg) # Notify network process and ground control
            self.running = False

elif msg.command == 'new_session' or msg.command == 'stop_session' or msg.command == 'set_gain':
95
96
97
                 self.fds.send(msg) # No housekeeping necessary, pass directly to spectrometer
98
            else:
                # Unknown command received from network
99
                  logging.warning('ctrl: unknown command: ' + msg.command)
```

7.1.3.4 def burn.burn.Burn.dispatch_spec_msg (self, msg)

Definition at line 102 of file burn.py.

```
def dispatch_spec_msg(self, msg):
    # Message received from spectrometer, pass on to network
    self.fdn.send(msg)
```

7.1.3.5 def burn.burn.Burn.run (self)

Entry point for the main controller

Definition at line 57 of file burn.py.

```
57
       def run(self):
58
            Entry point for the main controller
61
            self.running = True
62
           # Wait for gps to start up
63
            logging.info('ctrl: warming up services')
64
65
            time.sleep(5)
67
            \ensuremath{\text{\#}} Prepare file descriptors for selection
68
            inputs = [self.fdn, self.fds]
69
            # Main event loop
70
            while self.running:
                readable, _, _ = select.select(inputs, [], []) # Process readable file descriptors
72
73
                 for s in readable:
    msg = s.recv()
74
7.5
                     if s is self.fdn: # Message received from network
76
                         self.dispatch_net_msg(msg) # Handle network message
                     elif s is self.fds: # Message received from spectrometer
79
                          self.dispatch_spec_msg(msg) # Handle spectrometer message
80
```

7.1.4 Member Data Documentation

7.1.4.1 burn.burn.Burn.fdn

Definition at line 40 of file burn.py.

7.1.4.2 burn.burn.Burn.fds

Definition at line 39 of file burn.py.

7.1.4.3 burn.burn.Burn.n

Definition at line 48 of file burn.py.

7.1.4.4 burn.burn.Burn.running

Definition at line 36 of file burn.py.

7.1.4.5 burn.burn.Burn.s

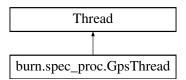
Definition at line 47 of file burn.py.

The documentation for this class was generated from the following file:

• /home/drb/dev/py/burn/burn.py

7.2 burn.spec_proc.GpsThread Class Reference

Inheritance diagram for burn.spec_proc.GpsThread:



Public Member Functions

- def __init__ (self, event)
- def run (self)
- def latitude (self)
- def epx (self)
- def longitude (self)
- def epy (self)
- def altitude (self)
- def epv (self)
- def speed (self)
- def eps (self)
- def time (self)

Public Attributes

- gpsd
- last_lat
- last_epx
- last_lon
- last_epy
- last_alt
- last_epv
- last_speed
- last_eps
- · last_time

7.2.1 Detailed Description

```
Thread class to handle the gps driver
```

Definition at line 38 of file spec proc.py.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 def burn.spec_proc.GpsThread.__init__ (self, event)

```
Description:
    Initialize the gps thread
```

Definition at line 42 of file spec_proc.py.

```
42
        def __init__(self, event):
43
             Description:
             Initialize the gps thread
47
             threading.Thread.__init__(self)
             self._stopped = event
self.gpsd = gps(mode=WATCH_ENABLE)
self.last_lat = 0
48
49
50
            self.last_epx = 0
51
            self.last_lon = 0
self.last_epy = 0
self.last_alt = 0
53
54
55
           self.last_epv = 0
self.last_speed = 0
56
57
             self.last_eps = 0
             self.last_time = ''
58
59
```

7.2.3 Member Function Documentation

7.2.3.1 def burn.spec_proc.GpsThread.altitude (self)

Definition at line 111 of file spec_proc.py.

```
111          def altitude(self):
112          return self.last_alt
113
```

```
7.2.3.2 def burn.spec_proc.GpsThread.eps ( self )
```

Definition at line 123 of file spec_proc.py.

```
123 def eps(self):
124 return self.last_eps
125
```

7.2.3.3 def burn.spec_proc.GpsThread.epv (self)

Definition at line 115 of file spec_proc.py.

```
115 def epv(self):
116 return self.last_epv
117
```

7.2.3.4 def burn.spec_proc.GpsThread.epx (self)

Definition at line 99 of file spec_proc.py.

```
99    def epx(self):
100        return self.last_epx
101
```

7.2.3.5 def burn.spec_proc.GpsThread.epy (self)

Definition at line 107 of file spec_proc.py.

```
107 def epy(self):
108 return self.last_epy
109
```

7.2.3.6 def burn.spec_proc.GpsThread.latitude (self)

Definition at line 95 of file spec_proc.py.

```
95 def latitude(self):
96 return self.last_lat
97
```

7.2.3.7 def burn.spec_proc.GpsThread.longitude (self)

Definition at line 103 of file spec_proc.py.

```
103 def longitude(self):
104 return self.last_lon
105
```

7.2.3.8 def burn.spec_proc.GpsThread.run (self)

```
Description:
    Entry point for the gps thread
```

Definition at line 60 of file spec_proc.py.

```
60
       def run(self):
61
62
           Description:
           Entry point for the gps thread
65
           logging.info('gps: starting service')
66
           \# Process any buffered gps signals every .3 seconds
67
68
           while not self._stopped.wait(0.3):
69
                # Update our last measurement until buffer is empty
71
               while self.gpsd.waiting():
72
                   self.gpsd.next()
                   if not math.isnan(self.gpsd.fix.latitude):
    self.last_lat = self.gpsd.fix.latitude
73
74
75
                   if not math.isnan(self.gpsd.fix.epx):
76
                       self.last_epx = self.gpsd.fix.epx
77
                    if not math.isnan(self.gpsd.fix.longitude):
78
                        self.last_lon = self.gpsd.fix.longitude
79
                    if not math.isnan(self.gpsd.fix.epy):
80
                        self.last_epy = self.gpsd.fix.epy
                    if not math.isnan(self.gpsd.fix.altitude):
                        self.last_alt = self.gpsd.fix.altitude
83
                    if not math.isnan(self.gpsd.fix.epv):
84
                        self.last_epv = self.gpsd.fix.epv
                    if not math.isnan(self.gpsd.fix.speed):
85
                        self.last_speed = self.gpsd.fix.speed
86
                    if not math.isnan(self.gpsd.fix.eps):
                        self.last_eps = self.gpsd.fix.eps
                    if self.gpsd.utc != None and self.gpsd.utc != '':
90
                        self.last_time = self.gpsd.utc
91
92
           logging.info('gps: terminating')
93
```

7.2.3.9 def burn.spec_proc.GpsThread.speed (self)

Definition at line 119 of file spec_proc.py.

```
119 def speed(self):
120 return self.last_speed
121
```

7.2.3.10 def burn.spec_proc.GpsThread.time (self)

Definition at line 127 of file spec_proc.py.

```
127 def time(self):
128 return self.last_time
```

7.2.4 Member Data Documentation

7.2.4.1 burn.spec_proc.GpsThread.gpsd

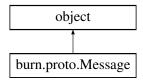
Definition at line 49 of file spec_proc.py.

7.2.4.2 burn.spec_proc.GpsThread.last_alt Definition at line 54 of file spec_proc.py. 7.2.4.3 burn.spec_proc.GpsThread.last_eps Definition at line 57 of file spec_proc.py. 7.2.4.4 burn.spec_proc.GpsThread.last_epv Definition at line 55 of file spec_proc.py. 7.2.4.5 burn.spec_proc.GpsThread.last_epx Definition at line 51 of file spec_proc.py. 7.2.4.6 burn.spec_proc.GpsThread.last_epy Definition at line 53 of file spec_proc.py. 7.2.4.7 burn.spec_proc.GpsThread.last_lat Definition at line 50 of file spec_proc.py. 7.2.4.8 burn.spec_proc.GpsThread.last_lon Definition at line 52 of file spec_proc.py. 7.2.4.9 burn.spec_proc.GpsThread.last_speed Definition at line 56 of file spec_proc.py. $7.2.4.10 \quad burn.spec_proc.GpsThread.last_time$ Definition at line 58 of file spec_proc.py. The documentation for this class was generated from the following file:

/home/drb/dev/py/burn/spec_proc.py

7.3 burn.proto.Message Class Reference

Inheritance diagram for burn.proto.Message:



Public Member Functions

def __init__ (self, command=", arguments={})

Public Attributes

- · command
- · arguments

7.3.1 Detailed Description

```
Class used to store a protocol message
```

Definition at line 20 of file proto.py.

7.3.2 Constructor & Destructor Documentation

```
7.3.2.1 def burn.proto.Message.__init__ ( self, command = ' ', arguments = { } )
```

Definition at line 24 of file proto.py.

Initialize message

7.3.3 Member Data Documentation

7.3.3.1 burn.proto.Message.arguments

Definition at line 29 of file proto.py.

7.3.3.2 burn.proto.Message.command

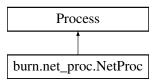
Definition at line 28 of file proto.py.

The documentation for this class was generated from the following file:

• /home/drb/dev/py/burn/proto.py

7.4 burn.net_proc.NetProc Class Reference

Inheritance diagram for burn.net_proc.NetProc:



Public Member Functions

- def __init__ (self, fd)
- def run (self)
- def dispatch_ctrl_msg (self, msg)
- def dispatch_net_msg (self)
- def is_running (self)

Public Attributes

- fd
- addr
- sock
- buffer

7.4.1 Detailed Description

Definition at line 28 of file net_proc.py.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 def burn.net_proc.NetProc.__init__ (self, fd)

```
Description:
    Initialization of the net process
Arguments:
    fd - File descriptor to send and receive messages to/from controller
```

Definition at line 30 of file net proc.py.

```
30
        def __init__(self, fd):
31
32
            Description:
33
                 Initialization of the net process
34
            Arguments:
            \tilde{\ } fd - File descriptor to send and receive messages to/from controller """
35
36
            Process.__init__(self)
self.fd = fd
37
39
            setblocking(self.fd, 0)
            self._running = False
self.conn, self.addr = None, None
self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
40
41
42
            self.sock.setblocking(0)
43
            self.buffer =
45
            try:
46
                 self.sock.bind((HOST, PORT))
            except socket.error as e:
47
                 logging.error('network: bind failed: ' + os.strerror(e.errno))
48
49
            self.sock.listen(5)
            logging.info('network: service listening')
52
```

7.4.3 Member Function Documentation

7.4.3.1 def burn.net_proc.NetProc.dispatch_ctrl_msg (self, msg)

```
Description:
    Serialize a controller message to a netstring and send it to ground control
Arguments:
    msg - The controller message to dispatch
```

Definition at line 109 of file net_proc.py.

```
def dispatch_ctrl_msg(self, msg):
109
110
111
             Description:
112
                  Serialize a controller message to a netstring and send it to ground control
113
             msg - The controller message to dispatch
114
115
             if msg.command == 'close_ok': # main controller is closing
116
                  self._running = False
117
118
119
             if msg.command == 'spectrum_ready': # Meta message
120
                     spectrum_ready' is a meta message indicating that a response message is stored on disk'
121
                  # Spectrum_leady is a meta message indicating that a respon
# The path to the file is stored in msg.arguments['filename']
with open(msg.arguments["filename"]) as jfd:
122
123
                  m = json.load(jfd) # Load json from file
data = json.dumps(m)
124
125
126
             else: # Regular message
127
                  data = json.dumps(msg.__dict__) # Convert object to json
128
129
              # Serialize the message into a netstring
130
             netstring = struct.pack("!I", len(data)) + data
              # Send the message over the network
```

```
132
           totlen, currlen = len(netstring), 0
133
           while True: # Continue until the full message is transferred
134
               1 = self.conn.send(netstring[currlen:])
135
               if 1 == 0:
136
                   inputs.remove(self.conn)
                   self.conn.close()
137
138
                   logging.info('network: connection broken from ' + self.addr[0])
139
               currlen += 1
140
               if currlen >= totlen:
141
142
143
```

7.4.3.2 def burn.net_proc.NetProc.dispatch_net_msg (self)

```
Description:
```

Convert received data to messages and pass them to the controller

Definition at line 144 of file net proc.py.

```
def dispatch_net_msg(self):
144
145
146
             Convert received data to messages and pass them to the controller
             Description:
148
149
             while True:
                 if len(self.buffer) < 4:</pre>
150
151
                  # Extract message length
152
                  msglen = struct.unpack("!I", self.buffer[0:4])[0]
153
                  if len(self.buffer) < msglen+4:</pre>
155
                       logging.info('network: buffer not ready')
156
                 # Extract rest of message and convert to object
jmsg = json.loads(self.buffer[4:4+msglen])
msg = Message(**jmsg)
157
158
159
160
                 # Pass message to main controller
161
                 self.fd.send(msg)
                 # Update buffer
self.buffer = self.buffer[4+msglen:]
162
163
164
```

7.4.3.3 def burn.net_proc.NetProc.is_running (self)

```
Description:
```

Return wether the net process is still running

Definition at line 165 of file net proc.py.

```
165 def is_running(self):
166 """
167 Description:
168 Return wether the net process is still running
169 """
170 return self._running
```

7.4.3.4 def burn.net_proc.NetProc.run (self)

```
Description:
Entry point for the net process
```

Definition at line 53 of file net proc.py.

```
def run(self):
53
55
           Description:
           Entry point for the net process
57
58
           logging.info('network: starting service')
           self._running = True
# Prepare sockets and file descriptors
59
60
           inputs = [self.fd, self.sock]
62
63
            # Start select event loop
           while(self._running):
    readable, _, _ = select.select(inputs, [], [])
64
65
66
                for s in readable: # Handle reads
68
                    if s is self.sock: # Incoming connection on listening socket
                        # We only allow one connection at a time (TODO)
self.conn, self.addr = s.accept()
69
70
71
                        self.conn.setblocking(0)
72
                        inputs.append(self.conn)
74
                        logging.info('network: connection received from ' + self.
      addr[0])
75
                    elif s is self.fd: # Incoming message from main controller
76
77
                        self.dispatch_ctrl_msg(s.recv())
78
79
                    else: # Incoming data from existing connection
80
                            data = s.recv(1024)
81
                        except socket.error as e:
82
                             if e.errno == errno.ECONNRESET:
83
                                 # Unexpected disconnect from client
                                 inputs.remove(s)
86
                                 s.close()
87
                             logging.error('network: ' + self.addr[0] + ': ' + os.strerror(e.errno))
88
                         if not data or data == '':
89
                             # Unexpected disconnect from client
90
                             inputs.remove(s)
92
                             s.close()
93
                             logging.error('network: connection lost')
94
95
96
                             # Data successfully received, store in buffer
                             self.buffer += data
98
                             self.dispatch_net_msg()
99
100
             # Close active connections
101
            if self.conn is not None:
102
                self.conn.close()
103
            if self.sock is not None:
104
                 self.sock.close()
105
            self.fd.close()
106
107
            logging.info('network: terminating')
108
```

7.4.4 Member Data Documentation

7.4.4.1 burn.net_proc.NetProc.addr

Definition at line 41 of file net_proc.py.

7.4.4.2 burn.net_proc.NetProc.buffer

Definition at line 44 of file net_proc.py.

7.4.4.3 burn.net_proc.NetProc.fd

Definition at line 38 of file net_proc.py.

7.4.4.4 burn.net_proc.NetProc.sock

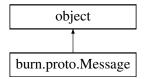
Definition at line 42 of file net_proc.py.

The documentation for this class was generated from the following file:

• /home/drb/dev/py/burn/net_proc.py

7.5 object Class Reference

Inheritance diagram for object:

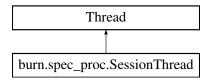


The documentation for this class was generated from the following file:

• /home/drb/dev/py/burn/proto.py

7.6 burn.spec_proc.SessionThread Class Reference

Inheritance diagram for burn.spec_proc.SessionThread:



Public Member Functions

- def __init__ (self, event, target, msg)
- def run (self)

7.6.1 Detailed Description

Thread class to govern a single session

Definition at line 130 of file spec_proc.py.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 def burn.spec_proc.SessionThread.__init__ (self, event, target, msg)

```
Description:
    Initialize the session thread
Arguments:
    event - Event to notify exit
    target - Function running the detector
    msg - The session message containing info about this session
```

Definition at line 134 of file spec proc.py.

```
def __init__(self, event, target, msg):
134
135
136
            Description:
137
                Initialize the session thread
138
            Arguments:
                event - Event to notify exit
139
                 target - Function running the detector
            msg - The session message containing info about this session
141
142
143
           threading.Thread.__init__(self)
           self._stopped = event
self._target = target
144
145
146
            self._msg = msg
```

7.6.3 Member Function Documentation

7.6.3.1 def burn.spec_proc.SessionThread.run (self)

```
Description:
Entry point for the session thread
```

Definition at line 148 of file spec_proc.py.

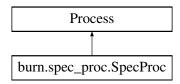
```
def run(self):
148
149
150
            Description:
            Entry point for the session thread
151
152
153
             logging.info('session: starting')
154
             # Extract the session variables from the session message
155
            delay = float(self._msg.arguments["delay"]) # Time to eait between each spectrum
            iterations = int(self._msg.arguments["iterations"]) # Number of spectrums to take infinite = iterations == -1 # If iterations is -1, run forever
156
157
            index = 0 # Keep track of spectrums (spectrum id)
158
159
160
             while not self._stopped.wait(delay):
161
                if not infinite:
162
                      # Exit when we reach a zero spectrum count
                      iterations = iterations - 1
163
164
                      if iterations < 0:</pre>
165
                 # Run the detector
166
167
                  self._target(self._msg, index)
168
                 index = index + 1
169
170
             logging.info('session: terminating')
```

The documentation for this class was generated from the following file:

/home/drb/dev/py/burn/spec_proc.py

7.7 burn.spec_proc.SpecProc Class Reference

Inheritance diagram for burn.spec_proc.SpecProc:



Public Member Functions

- def __init__ (self, fd)
- def run (self)
- def send_msg (self, msg)
- def dispatch (self, msg)
- def stabilize_probe (self, voltage, coarse_gain, fine_gain)
- def run_acquisition_once (self, req_msg, session_index)
- def reset_acquisition (self)
- def run_acquisition (self, msg, session_index)
- def save_acquisition (self, msg, session_index)
- def save_acquisition_as_chn (self, sd, msg, session_index)

Public Attributes

- fd
- running
- send_lock
- gps_stop
- gps_client
- group
- inputdtb
- session stop
- session

7.7.1 Detailed Description

Process class for handling the gps and spectrometry

Definition at line 172 of file spec_proc.py.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 def burn.spec_proc.SpecProc.__init__ (self, fd)

```
Description:
    Initialize the spectrometer process
Arguments:
    fd - file descriptor to pass and receive messages to/from controller
```

Definition at line 176 of file spec proc.py.

```
def __init__(self, fd):
176
177
178
             Description:
179
                  Initialize the spectrometer process
180
             Arguments:
             \bar{\mbox{fd}} – file descriptor to pass and receive messages to/from controller _{\mbox{\scriptsize mum}}
181
182
             Process.__init__(self)
self.fd = fd
183
184
185
             self.running = False
186
             self.send_lock = threading.Lock() # Lock used to syncronize sending of messages to
       controller
187
             self.gps_stop = threading.Event() # Event used to notify gps thread
             self.gps_client = GpsThread(self.gps_stop) # Create the gps thread
188
189
             # Initalize the detector
            self.input = 1
self.input = 1
190
191
192
             self.dtb = DeviceFactory.createInstance(DeviceFactory.DeviceInterface.IDevice)
             self.dtb.open("", Utilities.getLynxAddress())
logging.info('spec: using device ' + self.dtb.getParameter(ParameterCodes.Network_MachineName, 0))
193
194
             self.dtb.lock("administrator", "password", self.input)
195
```

7.7.3 Member Function Documentation

7.7.3.1 def burn.spec_proc.SpecProc.dispatch (self, msg)

```
Description:
   Handle a message received from controller
Arguments:
   msg - The message received
```

Definition at line 234 of file spec proc.py.

```
234
          def dispatch(self, msg):
235
236
               Description:
237
                   Handle a message received from controller
              Arguments:
               msg - The message received
239
240
              if msg.command == 'set_gain': # Set the gain parameters for the detector
  voltage = msg.arguments["voltage"]
  coarse = msg.arguments["coarse_gain"]
2.41
242
243
                    fine = msg.arguments["fine_gain"]
244
245
                    self.stabilize_probe(voltage, coarse, fine)
                    logging.info('spec: gain has been set')
msg.command = 'set_gain_ok' # Notify ground control that gain has been set
246
247
              self.send_msg(msg)
elif msg.command == 'close': # Controller wants us to close down
    self.running = False
248
249
250
251
               elif msg.command == 'new_session': # Start a new session
                   msg.command = 'new_session_ok'
252
253
                    self.send_msg(msg)
                    self.session_stop = threading.Event()
self.session = SessionThread(self.session_stop, self.
2.54
255
       run_acquisition_once, msq)
                    self.session.start()
```

```
elif msg.command == 'stop_session': # Stop any running sessions
258
                    if not self.session_stop.isSet():
259
                          self.session_stop.set()
                     self.session.join()
logging.info('spec: session stopped')
msg.command = 'stop_session_ok' # Notify ground control that we have stopped any sessions
260
2.61
262
263
                     self.send_msg(msg)
264
                else:
                     # Unknown command received from controller
logging.warning('spec: unknown command ' + cmd.command)
265
266
267
```

7.7.3.2 def burn.spec_proc.SpecProc.reset_acquisition (self)

```
Description:
Reset and initialize the detector
```

Definition at line 340 of file spec proc.py.

```
340
        def reset_acquisition(self):
341
342
            Description:
            Reset and initialize the detector
343
344
            #Disable all acquisition
346
            Utilities.disableAcquisition(self.dtb, self.input)
347
            #Set the acquisition mode. The Only Available Spectral in Osprey is Pha = 0
348
            self.dtb.setParameter(ParameterCodes.Input_Mode, 0, self.input)
349
            #Setup presets
350
            self.dtb.setParameter(ParameterCodes.Preset Options, 1, self.input)
351
            #Clear data and time
352
            self.dtb.control(CommandCodes.Clear, self.input)
353
            #Set the current memory group
354
            self.dtb.setParameter(ParameterCodes.Input_CurrentGroup, self.group, self.
      input)
355
```

7.7.3.3 def burn.spec_proc.SpecProc.run (self)

Definition at line 197 of file spec_proc.py.

```
197
        def run(self):
198
             Description:
199
             Entry point for the spectrometer process \ensuremath{\textbf{"""}}
200
201
202
             logging.info('spec: staring service')
203
             self.running = True
204
             self.gps_client.start() # Start the gps
205
206
207
             # Event loop
             while (self.running):
208
209
                 if self.fd.poll():
210
                      self.dispatch(self.fd.recv()) # Handle messages from the controller
211
             # Cleanup and exit
self.fd.close()
212
213
214
             self.gps_stop.set()
             self.gps_client.join()
216
             logging.info('spec: GPS client stopped')
             logging.info('spec: terminating')
217
218
```

7.7.3.4 def burn.spec_proc.SpecProc.run_acquisition (self, msg, session_index)

```
Description:
    Run the detector

Arguments:
    msg - The response message
    session_index - The sequence number in current session
```

Definition at line 356 of file spec_proc.py.

```
356
          def run_acquisition(self, msg, session_index):
357
358
               Description:
359
                    Run the detector
               Arguments:
360
                    msg - The response message
361
               session_index - The sequence number in current session
362
363
364
                # Setup presets
365
               livetime = float(msg.arguments["livetime"])
366
               self.dtb.setParameter(ParameterCodes.Preset_Live, livetime, self.input)
367
                # Clear data and time
               self.dtb.control(CommandCodes.Clear, self.input)
368
369
                # Start the acquisition
370
               self.dtb.control(CommandCodes.Start, self.input)
371
               while True:
372
                    sd = self.dtb.getSpectralData(self.input, self.group)
373
                    if ((0 == (StatusBits.Busy & sd.getStatus())) and (0 == (StatusBits.Waiting & sd.getStatus())))
374
375
                    time.sleep(.1)
376
377
                # Extract last spectrum from detector and prepare parameters
378
                chans = sd.getSpectrum().getCounts()
379
                total count = 0
                channel_string = ''
380
381
               for ch in chans:
382
                     total_count += ch
                     channel_string += str(ch) + ' '
383
384
               # Add spectrum data to the response message
msg.arguments["channels"] = channel_string.strip()
msg.arguments["channel_count"] = len(chans)
385
386
387
               msg.arguments["uncorrected_total_count"] = total_count
msg.arguments["livetime"] = sd.getLiveTime()
msg.arguments["realtime"] = sd.getRealTime()
388
389
390
               msg.arguments["reditime"] - Sd.getRearrime()
msg.arguments["computational_limit"] = sd.getComputationalValue()
msg.arguments["spectral_input"] = sd.getInput()
msg.arguments["spectral_group"] = sd.getGroup()
msg.arguments["spectral_status"] = Utilities.getStatusDescription(sd.getStatus())
391
392
393
394
395
```

7.7.3.5 def burn.spec_proc.SpecProc.run_acquisition_once (self, req_msg, session_index)

```
Description:
   Gather info from gps and detector

Arguments:
   req_msg - The session message
   session_index - The sequence number in current session
```

Definition at line 293 of file spec proc.py.

```
293 def run_acquisition_once(self, req_msg, session_index):
294 """
295 Description:
296 Gather info from gps and detector
297 Arguments:
298 req_msg - The session message
299 session_index - The sequence number in current session
300 """
301 # Prepare the response message
```

```
302
                   resp_msg = copy.deepcopy(req_msg)
303
                   resp_msg.command = 'spectrum
304
                   resp_msg.arguments['session_index'] = session_index
305
306
                   # Reset detector
307
                   self.reset_acquisition()
308
309
                   # Gather gps info before running the detector
                   resp_msg.arguments['latitude_start'] = self.gps_client.latitude
resp_msg.arguments['latitude_start_err'] = self.gps_client.epx
310
311
                   resp_msg.arguments['longitude_start'] = self.gps_client.longitude
312
                   resp_msg.arguments['longitude_start_err'] = self.gps_client.epy
313
                   resp_msg.arguments['altitude_start'] = self.gps_client.altitude
resp_msg.arguments['altitude_start'] = self.gps_client.altitude
resp_msg.arguments['altitude_start_err'] = self.gps_client.epv
314
315
                  resp_msg.arguments['gps_speed_start'] = self.gps_client.speed resp_msg.arguments['gps_speed_start_err'] = self.gps_client.eps resp_msg.arguments['gps_time_start'] = self.gps_client.time
316
317
318
319
                   # Run the detector
320
321
                   self.run_acquisition(resp_msg, session_index)
322
323
                   # Gather gps info after running the detector
                   resp_msg.arguments['gps_time_end'] = self.gps_client.time
resp_msg.arguments['latitude_end'] = self.gps_client.latitude
324
325
                   resp_msg.arguments['latitude_end_err'] = self.gps_client.epx
resp_msg.arguments['longitude_end'] = self.gps_client.longitude
326
327
                  resp_msg.arguments['longitude_end_err'] = self.gps_client.longitude_end_err'] = self.gps_client.epy resp_msg.arguments['altitude_end'] = self.gps_client.altitude resp_msg.arguments['altitude_end_err'] = self.gps_client.epv resp_msg.arguments['gps_speed_end'] = self.gps_client.speed
328
329
330
331
                  resp_msg.arguments['gps_speed_end_err'] = self.gps_client.eps
332
333
334
                   \# Save acquisition to file and send a meta message to controller
335
                   fn = self.save_acquisition(resp_msg, session_index)
                  m = Message('spectrum_ready')
m.arguments["filename"] = fn
336
337
338
                   self.send_msg(m)
```

7.7.3.6 def burn.spec_proc.SpecProc.save_acquisition (self, msg, session_index)

```
Description:
    Save the gps and specter data to file (json format)

Arguments:
    msg - The response message
    session_index - The sequence number in current session
```

Definition at line 396 of file spec proc.pv.

```
396
          def save_acquisition(self, msg, session_index):
397
398
               Description:
399
                    Save the gps and specter data to file (json format)
               Arguments:
400
                    msg - The response message
401
402
                     session_index - The sequence number in current session
403
               # Build the path to store the response message
session_name = msg.arguments['session_name']
session_dir = os.path.expanduser("~/ashes/") + session_name
if not os.path.isdir(session_dir):
404
405
406
407
               os.makedirs(session_dir, 0777)
fname = session_dir + os.path.sep + str(session_index) + ".json"
408
409
410
                # Store the response message to file
411
               with open(fname, "w") as f:
412
                     json.dump(msg.__dict__, f)
               return fname
413
414
```

7.7.3.7 def burn.spec_proc.SpecProc.save_acquisition_as_chn (self, sd, msg, session_index)

```
Description:
   Save the the specter data to file (chn format)
Arguments:
   sd - Spectrum data
   msg - The session message
   session_index - The sequence number in current session
```

Definition at line 415 of file spec_proc.py.

```
415
        def save_acquisition_as_chn(self, sd, msg, session_index):
416
417
           Description:
418
               Save the the specter data to file (chn format)
419
            Arguments:
420
             sd - Spectrum data
421
           session_index - The sequence number in current session
                msg - The session message
422
423
424
           session_name = msg.arguments['session_name']
425
           session_dir = os.path.expanduser("~/ashes/") + session_name
426
           if not os.path.isdir(session_dir):
427
                os.makedirs(session_dir, 0777)
428
           chans = sd.getSpectrum().getCounts()
      mca, sec, rt, lt, dat, tim, off, nc = 1, 0, sd.getRealTime(), sd.getLiveTime(), "07DEC151", "0707",
0, len(chans) # FIXME
429
430
           hdr = pack("hhhhii8s4shh", -1, mca, 1, sec, rt, lt, dat, tim, off, nc)
431
           with open(session_dir + os.path.sep + str(session_index) + ".chn", "w+b") as f:
432
                f.write(hdr)
                int_array = array('L', chans)
int_array.tofile(f)
433
434
435
```

7.7.3.8 def burn.spec_proc.SpecProc.send_msg (self, msg)

```
Description:
   Function to safely pass messages to controller Arguments:
   msg - The message to pass
```

Definition at line 219 of file spec_proc.py.

```
219
        def send_msg(self, msg):
220
221
            Description:
                Function to safely pass messages to controller
223
           msg - The message to pass
224
225
226
            self.send_lock.acquire()
227
           try:
228
               self.fd.send(msg)
           except:
230
                logging.error('spec: send exception: ' + sys.exc_info()[0])
231
            finall
232
                self.send_lock.release()
233
```

7.7.3.9 def burn.spec_proc.SpecProc.stabilize_probe (self, voltage, coarse_gain, fine_gain)

```
Description:
Set gain parameters for the detector Arguments:
voltage - The voltage level
coarse_gain - The coarse gain level
fine_gain - The fine gain level
```

Definition at line 268 of file spec_proc.py.

```
2.68
        def stabilize_probe(self, voltage, coarse_gain, fine_gain):
269
270
             Description:
                 Set gain parameters for the detector
272
            Arguments:
                 voltage - The voltage level
coarse_gain - The coarse gain level
fine_gain - The fine gain level
273
274
275
277
             # Osprey API constants
278
             Stabilized_Probe_Bussy = 0x00080000
             Stabilized_Probe_OK = 0x00100000
dtb_probe_type = self.dtb.getParameter(ParameterCodes.Input_Status, self.
279
280
      input)
281
282
             if((dtb_probe_type & Stabilized_Probe_OK) != Stabilized_Probe_OK):
283
                 self.dtb.setParameter(ParameterCodes.Input_Voltage, int(voltage), self.
      input)
284
                 self.dtb.setParameter(ParameterCodes.Input_VoltageStatus, True, self.
      input)
285
                 # Wait till ramping is complete
                 logging.info('spec: ramping HVPS...')
286
287
                 while (self.dtb.getParameter(ParameterCodes.Input_VoltageRamping, self.
      input) is True):
288
                     time.sleep(.4)
             # Set coarse and fine gain
289
             self.dtb.setParameter(ParameterCodes.Input_CoarseGain, float(coarse_gain), self.
290
      input) # [1.0, 2.0, 4.0, 8.0]
291
             self.dtb.setParameter(ParameterCodes.Input_FineGain, float(fine_gain), self.
      input) # [1.0, 5.0]
2.92
```

7.7.4 Member Data Documentation

7.7.4.1 burn.spec_proc.SpecProc.dtb

Definition at line 192 of file spec proc.py.

7.7.4.2 burn.spec_proc.SpecProc.fd

Definition at line 184 of file spec_proc.py.

7.7.4.3 burn.spec_proc.SpecProc.gps_client

Definition at line 188 of file spec_proc.py.

7.7.4.4 burn.spec_proc.SpecProc.gps_stop

Definition at line 187 of file spec_proc.py.

7.7.4.5 burn.spec_proc.SpecProc.group
Definition at line 190 of file spec_proc.py.
7.7.4.6 burn.spec_proc.SpecProc.input
Definition at line 191 of file spec_proc.py.
7.7.4.7 burn.spec_proc.SpecProc.running
Definition at line 185 of file spec_proc.py.
7.7.4.8 burn.spec_proc.SpecProc.send_lock
Definition at line 186 of file spec_proc.py.
7.7.4.9 burn.spec_proc.SpecProc.session
Definition at line 255 of file spec_proc.py.
7.7.4.10 burn.spec_proc.SpecProc.session_stop
Definition at line 254 of file spec_proc.py.
The documentation for this class was generated from the following file:

Chapter 8

File Documentation

8.1	/home/drb/dev/py/burn/_	_init_	py File Reference

• burn

Namespaces

8.2 /home/drb/dev/py/burn/burn.py File Reference

Classes

• class burn.burn.Burn

Namespaces

• burn.burn

Variables

- burn.burn.filename
- burn.burn.level

8.3 /home/drb/dev/py/burn/helpers.py File Reference

Namespaces

• burn.helpers

Functions

• def burn.helpers.setblocking (fd, state)

40 File Documentation

8.4 /home/drb/dev/py/burn/net_proc.py File Reference

Classes

class burn.net proc.NetProc

Namespaces

· burn.net proc

Variables

- string burn.net_proc.HOST = "
- int burn.net_proc.PORT = 7000

8.5 /home/drb/dev/py/burn/proto.py File Reference

Classes

• class burn.proto.Message

Namespaces

- burn.proto
- 8.6 /home/drb/dev/py/burn/README.md File Reference
- 8.7 /home/drb/dev/py/burn/spec_proc.py File Reference

Classes

- class burn.spec_proc.GpsThread
- class burn.spec_proc.SessionThread
- class burn.spec_proc.SpecProc

Namespaces

• burn.spec proc

Index

/home/drb/dev/py/burn/README.md, 40	fds, 17
/home/drb/dev/py/burn/initpy, 39	n, 18
/home/drb/dev/py/burn/burn.py, 39	run, 17
/home/drb/dev/py/burn/helpers.py, 39	running, 18
/home/drb/dev/py/burn/net_proc.py, 40	s, 18
/home/drb/dev/py/burn/proto.py, 40	
/home/drb/dev/py/burn/spec_proc.py, 40	burn::helpers
enter	setblocking, 12
	burn::net_proc
burn::burn::Burn, 16	HOST, 12
exit	PORT, 12
burn::burn::Burn, 16	burn::net_proc::NetProc
init	init, 25
burn::burn::Burn, 15	addr, 27
burn::net_proc::NetProc, 25	buffer, 27
burn::proto::Message, 23	dispatch_ctrl_msg, 25
burn::spec_proc::GpsThread, 19	dispatch_net_msg, 26
burn::spec_proc::SessionThread, 29	fd, 27
burn::spec_proc::SpecProc, 31	is_running, 26
	run, 26
addr	sock, 28
burn::net_proc::NetProc, 27	burn::proto::Message
altitude	init, 23
burn::spec_proc::GpsThread, 19	arguments, 23
arguments	command, 23
burn::proto::Message, 23	burn::spec_proc::GpsThread
	init, 19
buffer	altitude, 19
burn::net_proc::NetProc, 27	
burn, 11	eps, 19
burn.burn, 11	epv, 20
burn.burn.Burn, 15	epx, 20
burn.helpers, 12	epy, 20
burn.net_proc, 12	gpsd, 21
burn.net_proc.NetProc, 24	last_alt, 21
burn.proto, 13	last_eps, 22
burn.proto.Message, 23	last_epv, 22
burn.spec_proc, 13	last_epx, 22
burn.spec proc.GpsThread, 18	last_epy, 22
burn.spec_proc.SessionThread, 28	last_lat, 22
burn.spec_proc.SpecProc, 30	last_lon, 22
burn::burn	last_speed, 22
filename, 11	last_time, 22
level, 11	latitude, 20
burn::burn::Burn	longitude, 20
enter, 16	run, 20
exit, 16	speed, 21
exit, 16 init, 15	time, 21
dispatch_net_msg, 16	burn::spec_proc::SessionThrea
dispatch_spec_msg, 17	init, 29
. – . – .	
fdn, 17	run, <mark>29</mark>

42 INDEX

burn::spec_proc::SpecProc	burn::spec_proc::SpecProc, 36
init, 31	gpsd
dispatch, 31	burn::spec_proc::GpsThread, 21
dtb, 36	group
fd, 36	burn::spec_proc::SpecProc, 36
gps_client, 36	
gps_stop, 36	HOST
group, 36	burn::net_proc, 12
input, 37	
reset_acquisition, 32	input
run, 32	burn::spec_proc::SpecProc, 37
run_acquisition, 32	is_running
run_acquisition_once, 33	burn::net_proc::NetProc, 26
running, 37	L. A. H.
save_acquisition, 34	last_alt
save_acquisition_as_chn, 34	burn::spec_proc::GpsThread, 21
send_lock, 37	last_eps
send_msg, 35	burn::spec_proc::GpsThread, 22
session, 37	last_epv
session_stop, 37	burn::spec_proc::GpsThread, 22
stabilize probe, 35	last_epx
Stabilize_probe, 33	burn::spec_proc::GpsThread, 22
command	last_epy
burn::proto::Message, 23	burn::spec_proc::GpsThread, 22
burnprotoiwessage, 23	last_lat
dispatch	burn::spec_proc::GpsThread, 22
•	last_lon
burn::spec_proc::SpecProc, 31	burn::spec_proc::GpsThread, 22
dispatch_ctrl_msg	last_speed
burn::net_proc::NetProc, 25	burn::spec_proc::GpsThread, 22
dispatch_net_msg	last_time
burn::burn::Burn, 16	burn::spec_proc::GpsThread, 22
burn::net_proc::NetProc, 26	latitude
dispatch_spec_msg	burn::spec_proc::GpsThread, 20
burn::burn::Burn, 17	level
dtb	burn::burn, 11
burn::spec_proc::SpecProc, 36	longitude
	burn::spec_proc::GpsThread, 20
eps - C T - L 10	
burn::spec_proc::GpsThread, 19	n
epv	burn::burn::Burn, 18
burn::spec_proc::GpsThread, 20	
ерх	object, 28
burn::spec_proc::GpsThread, 20	
еру	PORT
burn::spec_proc::GpsThread, 20	burn::net_proc, 12
fd	reset_acquisition
burn::net_proc::NetProc, 27	burn::spec_proc::SpecProc, 32
burn::spec_proc::SpecProc, 36	run
fdn	burn::burn::Burn, 17
burn::burn::Burn, 17	burn::net_proc::NetProc, 26
fds	burn::spec_proc::GpsThread, 20
burn::burn::Burn, 17	burn::spec_proc::SessionThread, 29
filename	burn::spec_proc::SpecProc, 32
burn::burn, 11	run_acquisition
	burn::spec_proc::SpecProc, 32
gps_client	run_acquisition_once
burn::spec_proc::SpecProc, 36	burn::spec_proc::SpecProc, 33
gps_stop	running

INDEX 43

```
burn::burn::Burn, 18
    burn::spec_proc::SpecProc, 37
s
    burn::burn::Burn, 18
save_acquisition
    burn::spec_proc::SpecProc, 34
save_acquisition_as_chn
    burn::spec_proc::SpecProc, 34
send_lock
    burn::spec_proc::SpecProc, 37
send msg
    burn::spec_proc::SpecProc, 35
session
    burn::spec_proc::SpecProc, 37
session_stop
    burn::spec_proc::SpecProc, 37
setblocking
    burn::helpers, 12
sock
    burn::net_proc::NetProc, 28
speed
    burn::spec_proc::GpsThread, 21
stabilize_probe
    burn::spec_proc::SpecProc, 35
time
    burn::spec_proc::GpsThread, 21
```